

CPU 19

QUADERNO DI AGGIORNAMENTO TECNICO

Codice Pubblicazione : 920,45,1

Codice Aggiornamento : 01

Agosto 1977

INDICE

- Premessa - Prestazioni - Microprogrammi e firmware	Pag. 43,01
- Scratch-pad	" 43,02
- Registro deviatori	" 43,04
- Memorie per CPU 19	" 43,05
- Mappa di memoria del P6060	" 43,11
- Microprogram counter - Interfaccia verso periferiche	" 43,16
- Microprogrammi di gestione delle interruzioni	" 43,20
- Suddivisione delle microistruzioni per funzione	" 43,31

PREMESSA

Il CPU 19 è un "processor" realizzato dall'OLIVETTI ed è utilizzato su una vasta gamma di prodotti: TC 800, P 6060, DE 700 ed altri a venire.

L'intero "processor" è realizzato su due piastre standard a micrologici : PUCE 1 e PUCE 2.

PRESTAZIONI

Il CPU 19 è in grado di eseguire un set di 126 microistruzioni diverse che operano con :

- registri interni al CPU 19, realizzati in hardware e detti "Scratch-pad". Lo scratch-pad ha una capacità totale di 256 bit.
- un registro deviatori di 8 bit.
- memoria esterna (fino a 64 KW di 16 bit).
- governi di unità periferiche (GOP).
- console ausiliaria per riparazioni e collaudi : SUCE 2.

Ogni microistruzione è codificata con 16 bit (una word) e può essere inviata al CPU 19 da ROM, RAM o da SUCE 2.

In appendice è riportata una tabella riassuntiva delle microistruzioni che il CPU 19 può eseguire.

MICROPROGRAMMI E FIRMWARE

Le microistruzioni sono istruzioni a livello di elaborazione molto basso, operano infatti tra byte e al massimo una microistruzione tratta una word (16 bit).

Con un insieme di microistruzioni ordinate (microprogrammi) è possibile però trattare degli elementi più complessi di una singola word.

Esempio :

- numeri decimali
- campi di dati numerici e alfanumerici
- I/O di campi di memoria su periferiche esterne.

Quindi se il CPU 19 è dotato di una serie di opportuni microprogrammi diventa una macchina di livello superiore in grado di eseguire delle istruzioni di livello molto più elevato che non le singole micro-istruzioni.

Il corredo di microprogrammi che accompagnano il CPU 19 sono detti FIRMWARE della nuova macchina costituita da CPU 19 + microprogrammi.

E' chiaro che realizzando diversi FIRMWARE di corredo al CPU 19 si possono avere macchine dalle prestazioni molto diverse.

SCRATCH - PAD

Lo scratch-pad (fig. 1) è organizzato in 16 registri di 16 bit chiamati $L\emptyset + LF$.

Ognuno dei 16 registri L può essere diviso in due registri di 8 bit ognuno chiamati A e B.

Si possono così ottenere :

- 16 registri A, corrispondenti ai 16 byte meno significativi dei registri $L\emptyset + LF$ e denominati $A\emptyset + AF$;
- 16 registri B, (corrispondenti ai 16 byte più significativi dei registri $L\emptyset + LF$) denominati $B\emptyset + BF$.

Alcune microistruzioni particolari possono operare anche solo su metà dei registri A e B : in questo caso i quattro bit meno significativi (da $\emptyset$ a 3) di ogni registro A o B vengono chiamati rispettivamente AM (-) e BM (-); mentre i quattro bit più significativi vengono chiamati rispettivamente AP (+) e BP (+).

Il microprogrammatore può a sua scelta indirizzare i vari registri L, B, A, AP (+), AM (-), BP (+), BM (-); non può indirizzare i singoli bit.

Si possono infatti avere microistruzioni del tipo :

- a) ICL Lx : che incrementa di 1 l'intero contenuto del registro L selezionato.
- b) SAB Ax By : che scambia fra di loro i contenuti di un qualsiasi registro A (Ax) e un qualsiasi registro B (By).
- c) AZAP Ax : che azzerà i 4 bit della parte più significativa di un registro A [AP (+)] .

SCRATCH PAD											
B						A					
	7	4	3	0		7	4	3	0		
L 0											
L 1											
L 2											
L 3											
L 4											
L 5											
L 6											
L 7											
L 8											
L 9											
L A											
L B											
L C											
L D											
L E											
L F											
BP(+) BM(-)						AP(+) AM(-)					

Fig. 1

REGISTRO DEVIATORI

Il registro deviatori "D" comprende 8 bit chiamati D0+D7 (fig. 2).

Esistono alcune microistruzioni per il trattamento dell'intero registro deviatori. In particolare però i deviatori D0, D1 e D2 sono singolarmente posizionati da alcune microistruzioni che memorizzano su di essi il risultato qualitativo dell'operazione.

Esempio :

- nel caso di somma tra un registro A ed un registro B di scratch-pad, il deviatore D0 va in set ogni volta che il risultato è maggiore di 255; mentre è messo a zero se il risultato è inferiore a 255.
- nel caso di AND fra un registro A e B, il deviatore D1 è messo in set se il risultato dà 8 bit a zero; mentre sarà sempre messo a zero se il risultato dà almeno un bit ad 1.

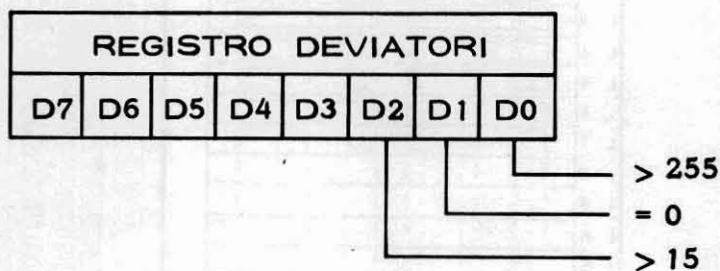


Fig. 2

I singoli deviatori possono essere sondati dalle microistruzioni di salto condizionato e condizionare così lo svolgimento dei microprogrammi.

MEMORIE PER CPU 19

Capacità e tipo di memoria

Il CPU 19 può gestire direttamente un massimo di 64 KWORD (16 bit) di memoria viva.

La memoria può essere un insieme di memorie di tipo diverso: RAM, ROM, PROM e ogni modulo può avere un tempo d'accesso diverso purchè rispetti l'interfaccia prevista dal CPU 19.

Nella memoria viva possono essere memorizzati microprogrammi, programmi e dati.

Indirizzi della memoria a livello word

La memoria viva del P 6060 è costituita da un pacco massimo di 64 KW (chiloword) - fig. 3.

- I 64 KW possono essere suddivisi in 16 moduli (da 0 a F) di 4 KW ognuno.
- Ciascun modulo di 4 KW può essere suddiviso in 16 pagine (da 0 a F) di 256 word ognuna.
- Ciascuna pagina contiene 256 word (da 00 ad FF).

L'indirizzo di memoria che il CPU 19 deve emettere per selezionare una word, deve avere 16 bit (BC00 + BC15), di cui :

- BC00 + BC07 (due cifre esadecimali : $X_2 X_1$) selezionano una word nell'ambito di ogni pagina.
- BC08 + BC11 (una cifra esadecimale : Y) selezionano una delle 16 pagine nell'ambito di un modulo di memoria di 4Kword.
- BC12 + BC15 (una cifra esadecimale : Z) selezionano nell'ambito dell'intera memoria un modulo di 4Kword.

I 16 bit d'indirizzo possono essere memorizzati in un registro $L(x)$ di Scratch-pad.

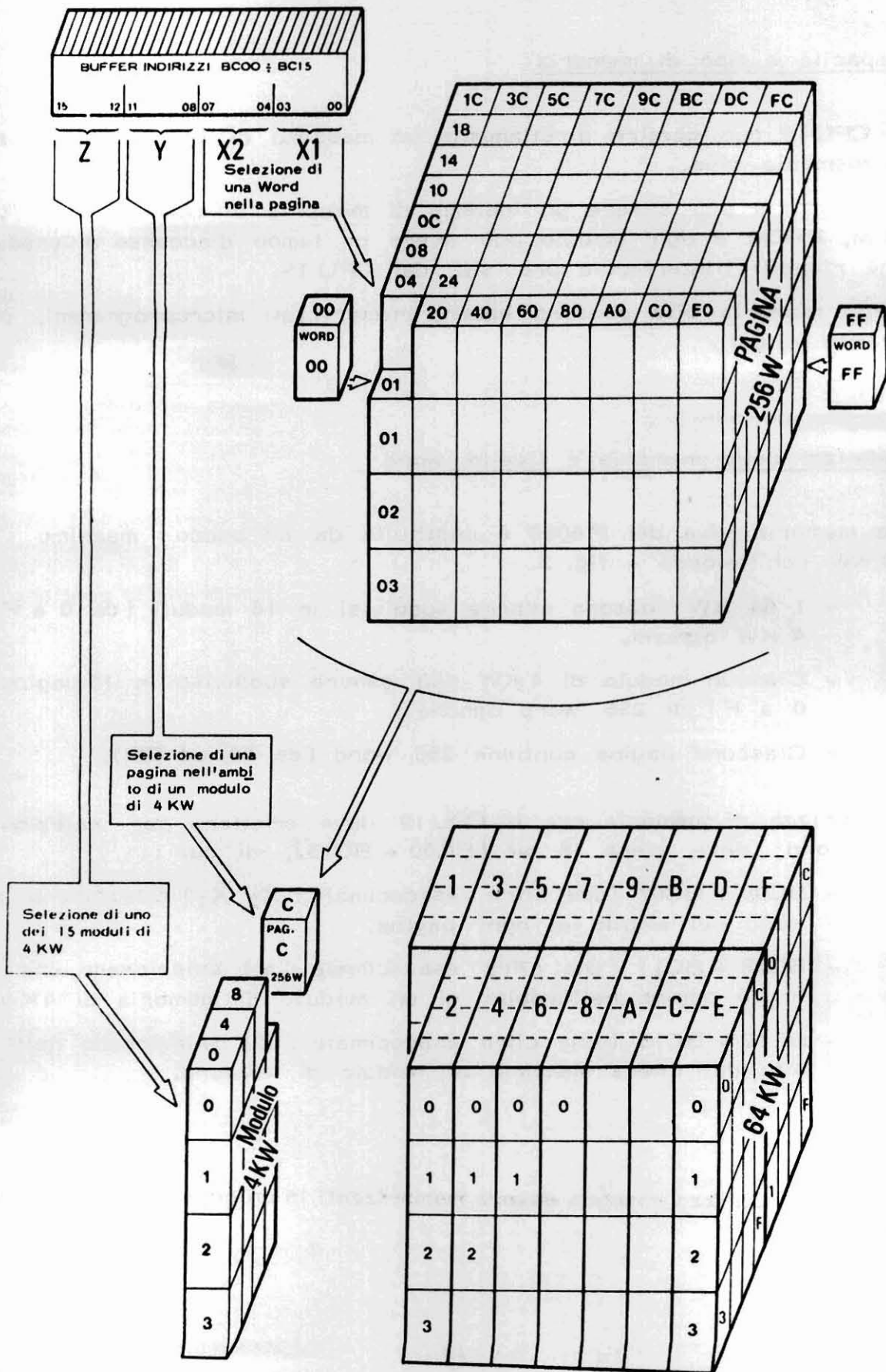


Fig. 3

Indirizzi di memoria a livello byte

L'accesso alle word di memoria è indispensabile quando si vuol leggere una microistruzione o quando si vogliono trasferire dati o indirizzi da o su registri $L(x)$ di Scratch-pad.

Molte microistruzioni prevedono invece lo scambio dati tra CPU 19 e memorie a livello byte (8 bit).

La memoria è stata per questo suddivisa anche in byte e di seguito esamineremo appunto l'organizzazione della memoria in byte.

Ogni word di 16 bit è divisa in due byte (fig. 4 e 5):

- il byte meno significativo (-) che comprende bit da 0 a 7;
il cui indirizzo è il doppio +1 dell'indirizzo di word che lo contiene
- il byte più significativo (+) che comprende i bit da 8 a F.
il cui indirizzo è il doppio dell'indirizzo di word che lo contiene.

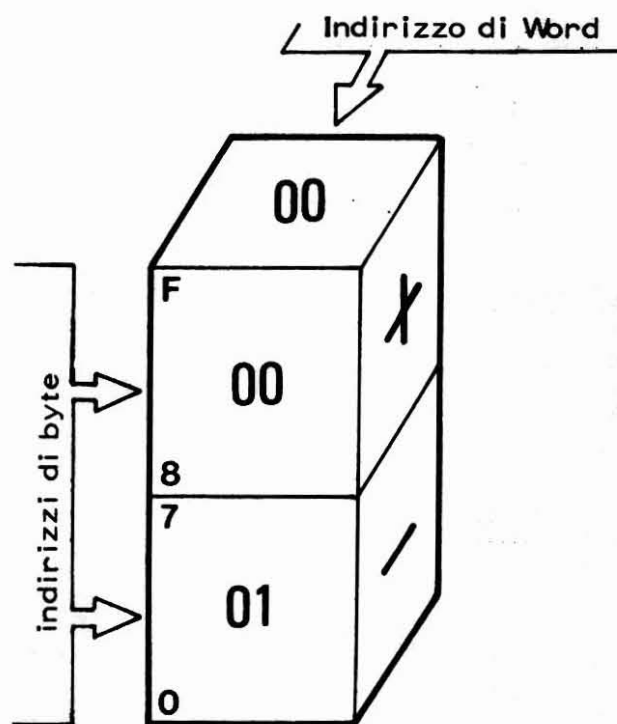


Fig. 4

Per poter indirizzare uno ad uno i 128 Kbyte possibili nell'intero pacco di memoria, occorrerebbero 17 bit di indirizzo.

Nello Scratch-pad non ci sono registri di 17 bit: perciò le micro-istruzioni che operano con memoria a livello byte possono indirizzare solo la prima metà del pacco di memoria; cioè la memoria contenuta tra le word 0000 e 7FFF ovvero i byte 0000+FFFF (fig. 6).

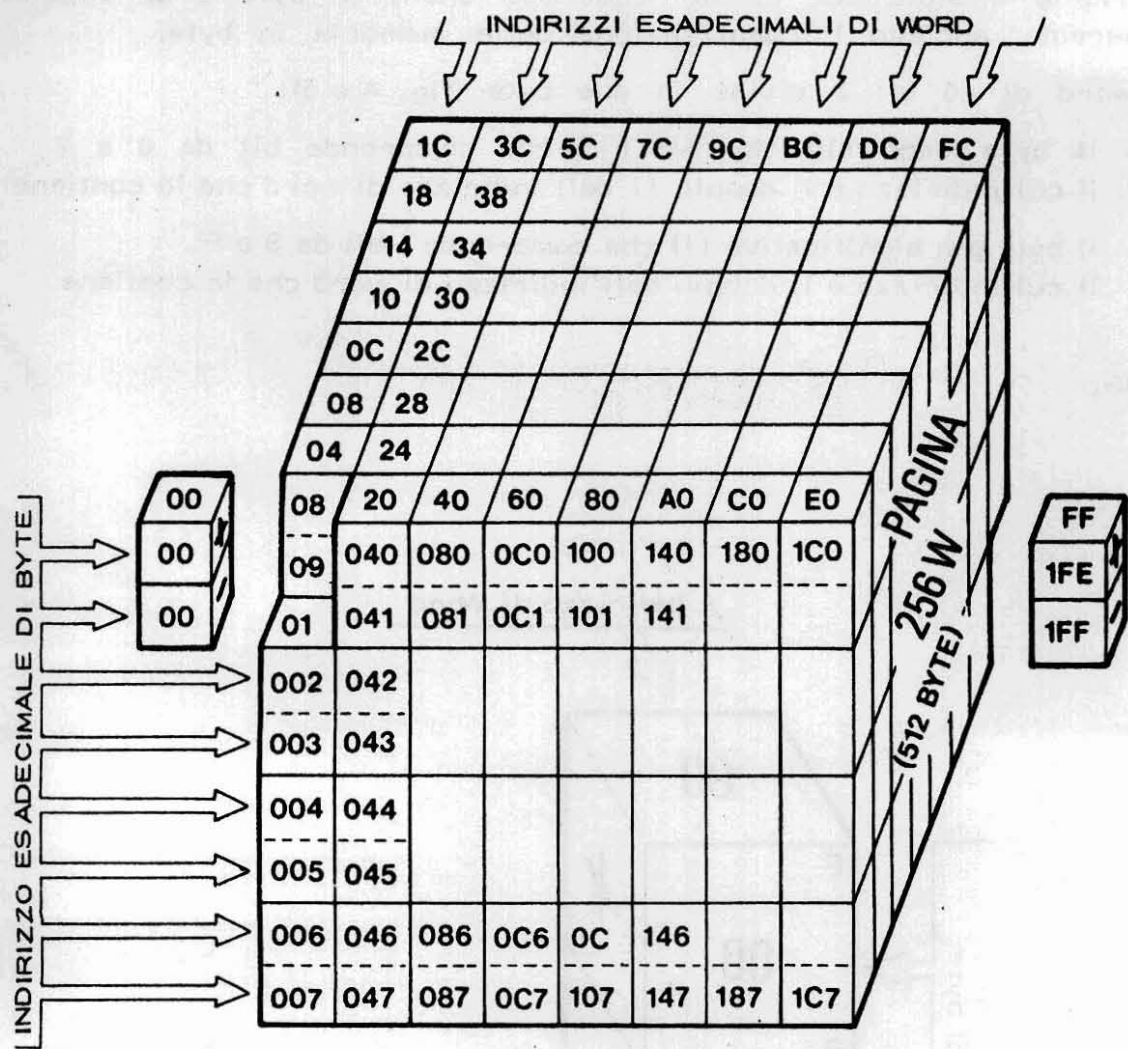


Fig. 5

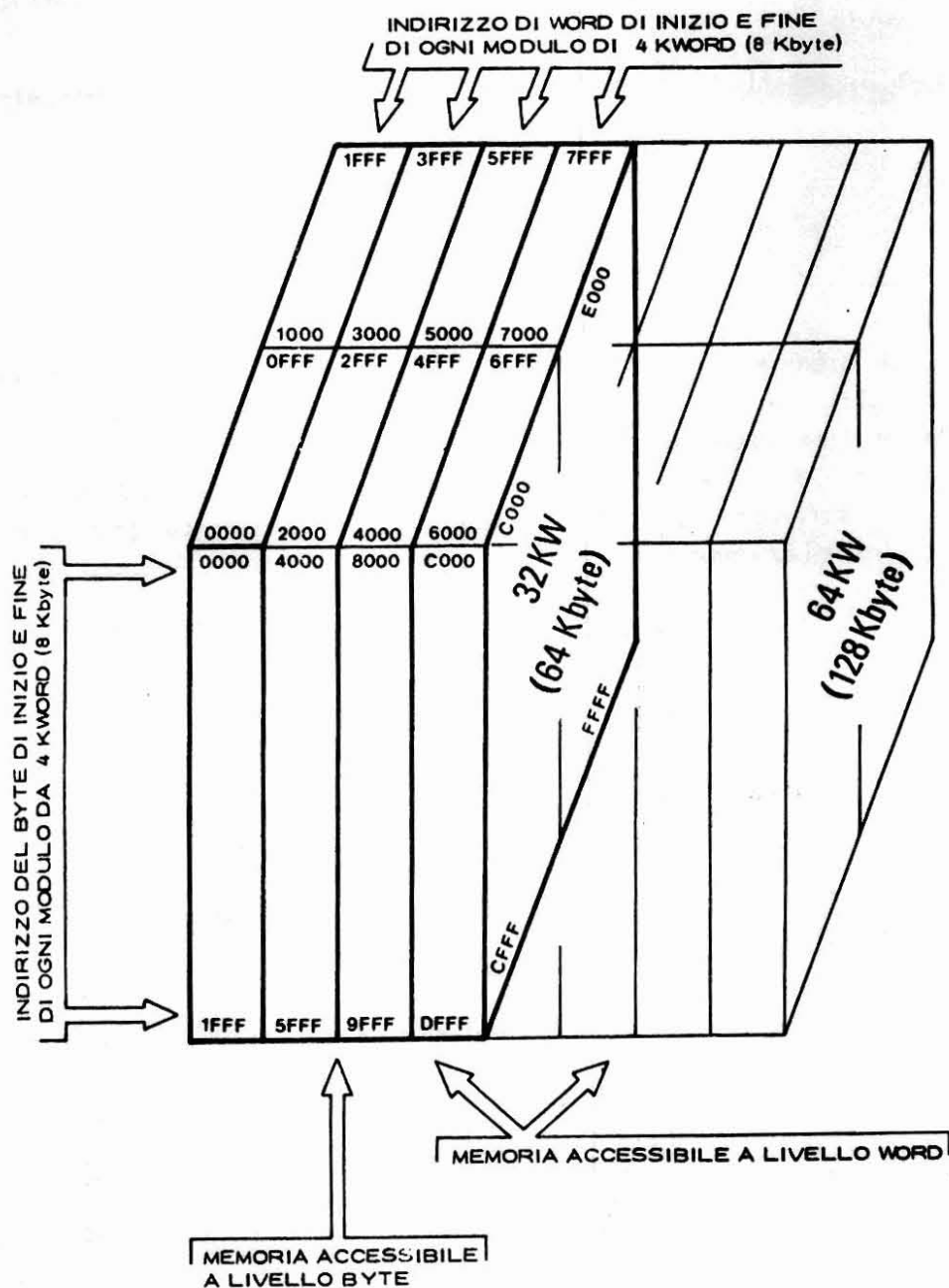


Fig. 6

Relazione tra indirizzo di byte e di word

Se si divide l'indirizzo di un byte per due si hanno :

- risultato uguale all'indirizzo della word che contiene il byte;
- resto 0 se il byte corrisponde al byte più significativo della word;
- resto 1 se il byte corrisponde al byte meno significativo della word.

Composizione del pacco di memoria

I 64 Kword di memoria possibili, non sempre sono tutti utilizzati. Di solito si riservano le word comprese tra gli indirizzi 8000 e FFFF per le memorie destinate al firmware che richiede solo un indirizzamento per word mentre le word comprese tra gli indirizzi 0000 e 7FFF sono destinate al software di base e alla memoria utente che richiedono il duplice indirizzamento: per byte e per word.

La memoria è in genere assemblata su un certo numero di piastre di capacità e di tipo diverso.

A titolo di esempio, formiamo, in figura 7, una mappa di memoria del P6060 in configurazione minima.

Come si può ben vedere, nello stesso pacco di memoria sono inserite indifferentemente : ROM o PROM e diverse RAM.

La RAM a sua volta può essere costituita da più tipi di piastre tra cui citiamo le seguenti :

Nome commerciale	Sigla della piastra	Capacità
MEM 8003	RAMA 8 B	8 K word
MEM 2208	RAMA 8 C	8 K word
MEM 2132	RAMA 16	16 K word

MAPPA DI MEMORIA DEL P 6060

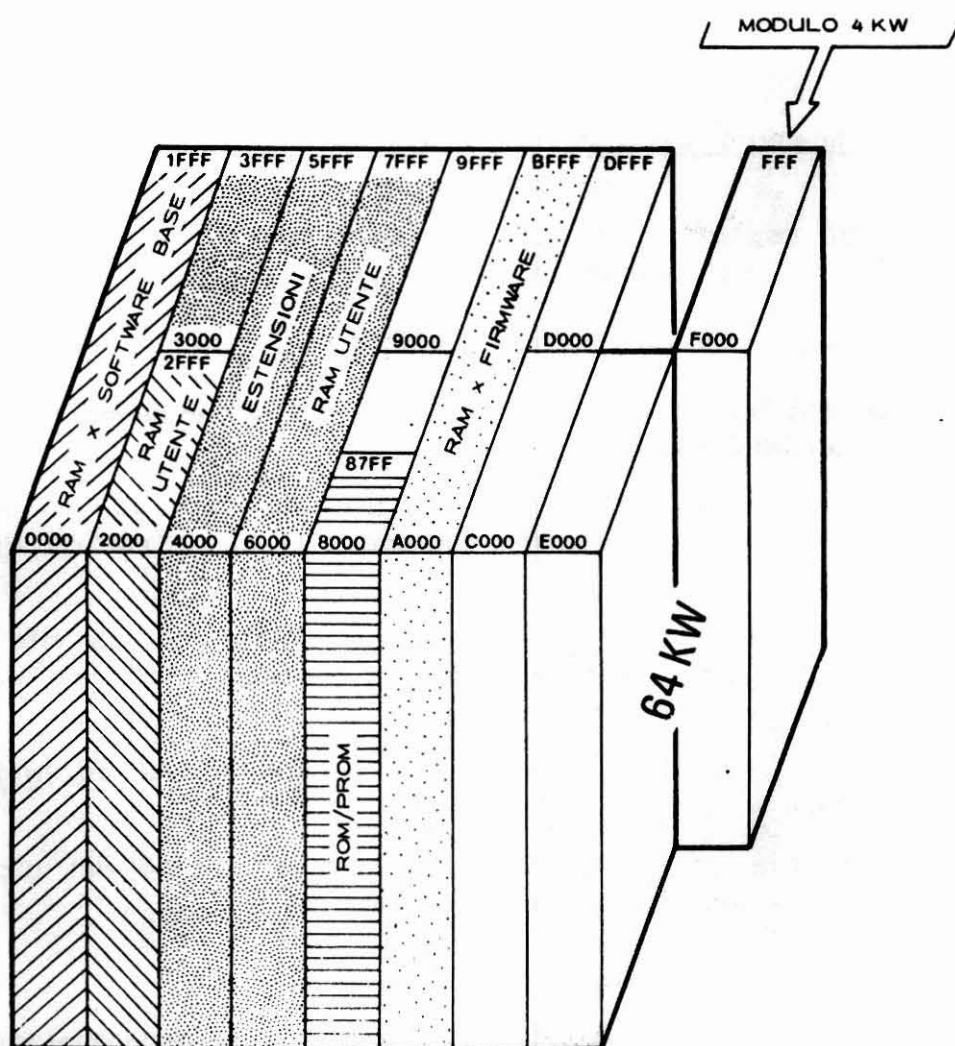


Fig. 7

Tempo di accesso alla memoria

Il colloquio memoria-CPU19 è di tipo asincrono: il CPU19 lancia un ciclo di lettura o scrittura, la piastra di memoria che contiene la cella indirizzata si prende tutto il tempo che le occorre per completare il ciclo e comunica all'U.C. la fine del ciclo.

Il tempo di accesso alla memoria è quindi variabile ed è legato alla piastra di memoria usata.

Procedura di scambio dati tra CPU19 e memoria

Vedremo qui di seguito la procedura usata per lo scambio dati tra CPU19 e memoria nei 4 casi significativi.

1) Scrittura di una word in memoria (fig. 8).

L'indirizzo di word (nell'esempio: 17 AC) viene memorizzato sul buffer BC00 + 15.

Il segnale WBCON che carica gli indirizzi sul buffer BC è inviato alla memoria per indicare che ha inizio un ciclo di memoria.

La presenza dei segnali WMBP e WMBM indica che si vuole avviare un ciclo di scrittura sia nel byte "+" che nel byte "-" della word indirizzata.

Il dato da scrivere è memorizzato su CPU19 ed inviato sul bus ECD0 + ECDF.

La memoria scrive il dato e comunica la fine del ciclo di scrittura emettendo il segnale MEOCN.

2) Scrittura di un byte in memoria (fig. 9).

L'indirizzo di byte predisposto dal microprogrammatore nell'esempio 0015) viene diviso per due :

- il risultato, indirizzo di word, è memorizzato sul buffer BC ed è inviato alla memoria ;
- il resto della divisione abilita la nascita del segnale WMBM se 1 o del segnale WMBP se 0, i quali stanno ad indicare che si vuole eseguire un ciclo di scrittura ed in particolare si vuole scrivere nel solo byte (-) se WMBMN = 0 ; si vuole scrivere nel solo byte (+) se WMBPN = 0

SCRITTURA DI UNA WORD IN MEMORIA

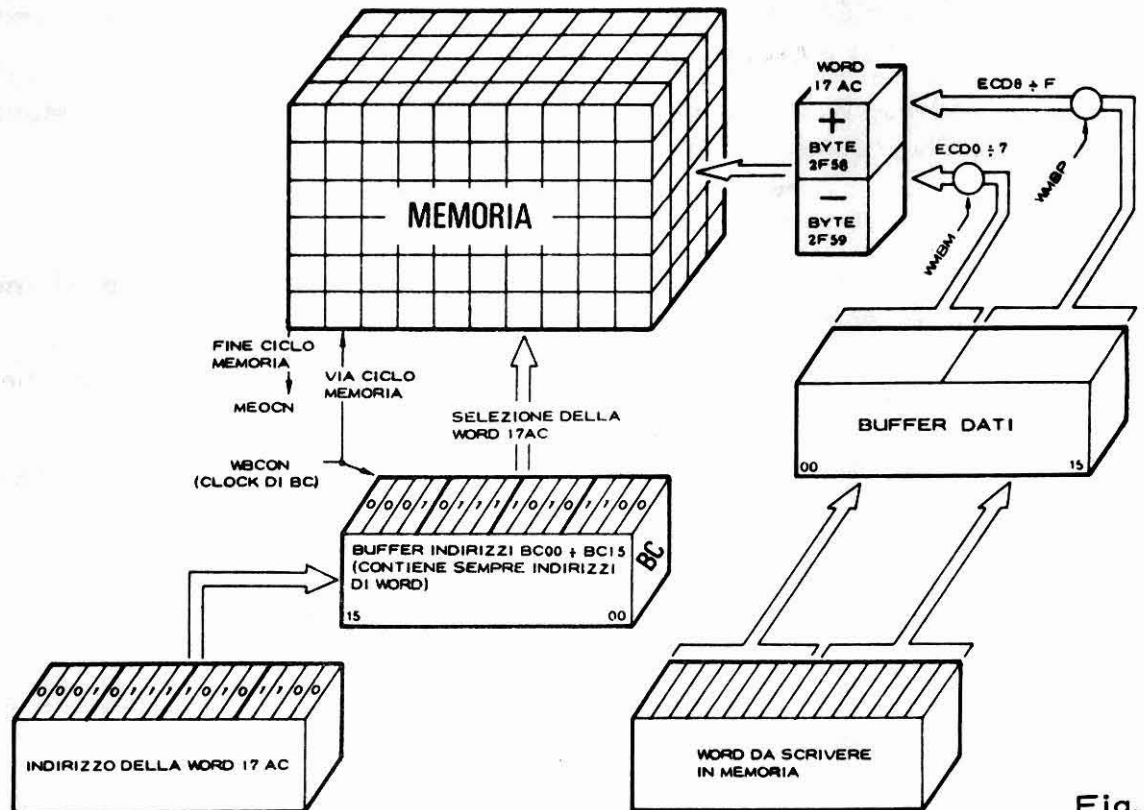


Fig. 8

SCRITTURA DI UN BYTE IN MEMORIA

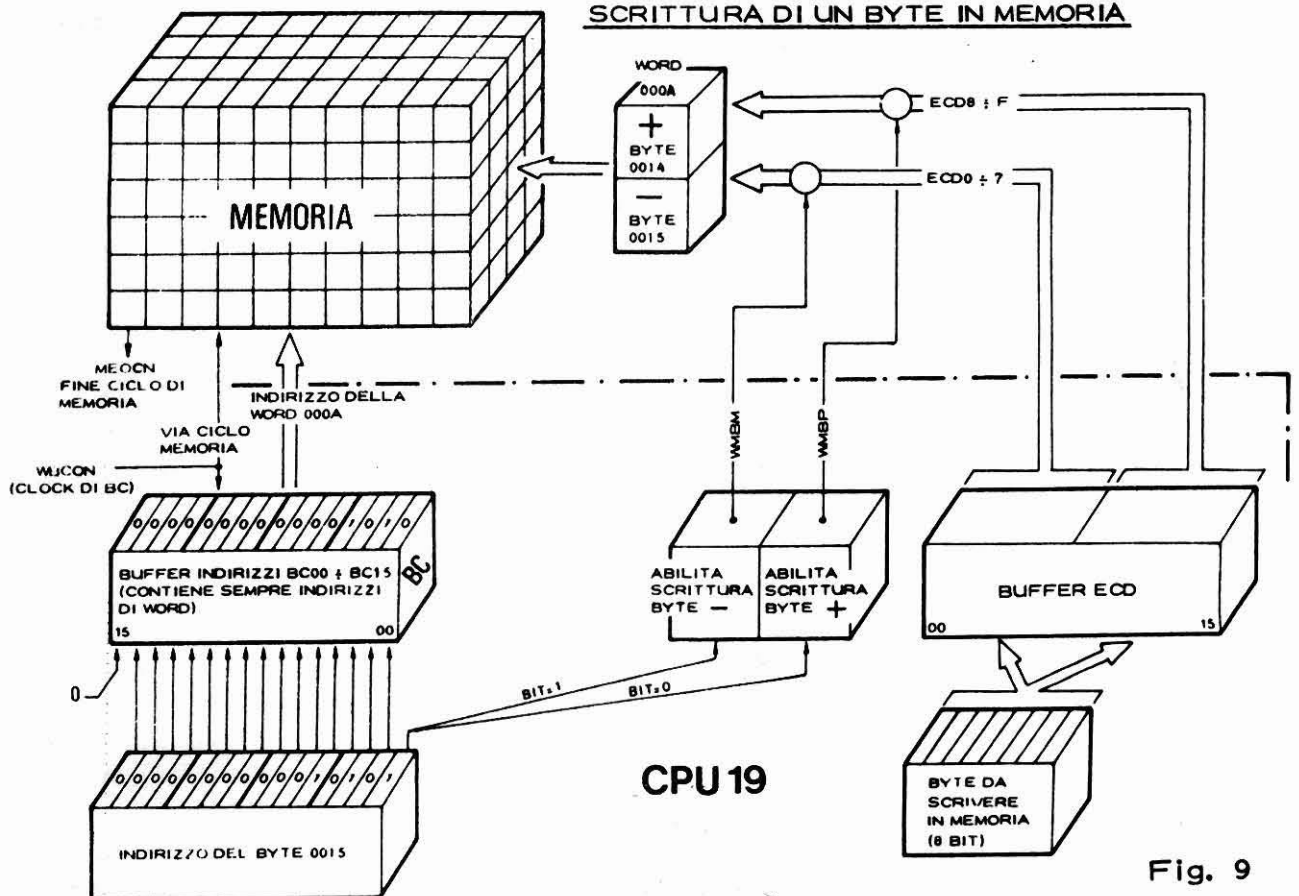


Fig. 9

Il byte da scrivere in memoria è presentato sulle due metà del bus ECD e solo la parte del bus ECD abilitata verrà scritta nella word.

Il nastro WBCON dà l'inizio del ciclo di memoria mentre MEOCN comunica all'U.C. la fine del colloquio.

3) Lettura di una word.

Un ciclo di lettura è avviata col segnale WBCON e l'indirizzo è emesso dal buffer BC.

L'assenza dei segnali WMBP e WMBM sta ad indicare che si vuole un ciclo di lettura.

La word letta è emessa verso CPU19 sul bus ME000+ME150.

La memoria comunica che i dati emessi sono validi emettendo il segnale MEOCN.

4) Lettura di un byte (fig. 11).

L'indirizzo del byte predisposto dal microprogrammatore è diviso per due e:

- il risultato (indirizzo di word) è emesso verso memoria sul bus BC000+BC150;
- il resto seleziona uno dei due byte della word emessa dalla memoria su ME000+ME15.

Il segnale WBCON comunica alla memoria l'avvio del ciclo e l'assenza dei segnali WMBP e WMBM sta ad indicare che il ciclo voluto è di lettura.

La memoria comunica la validità dei dati sul bus ME000+15 col segnale MEOCN.

LETTURA DELLA WORD DI MEMORIA 017C

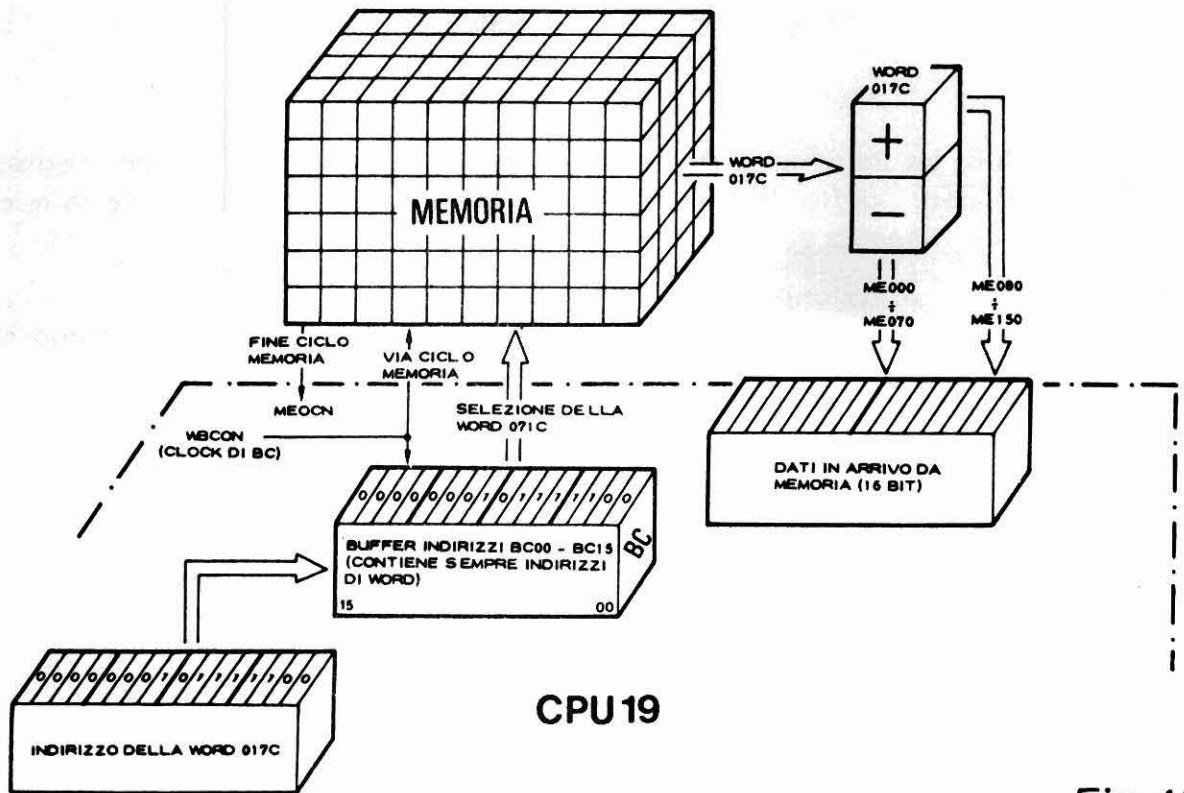


Fig. 10

LETTURA DEL BYTE DI MEMORIA 0007

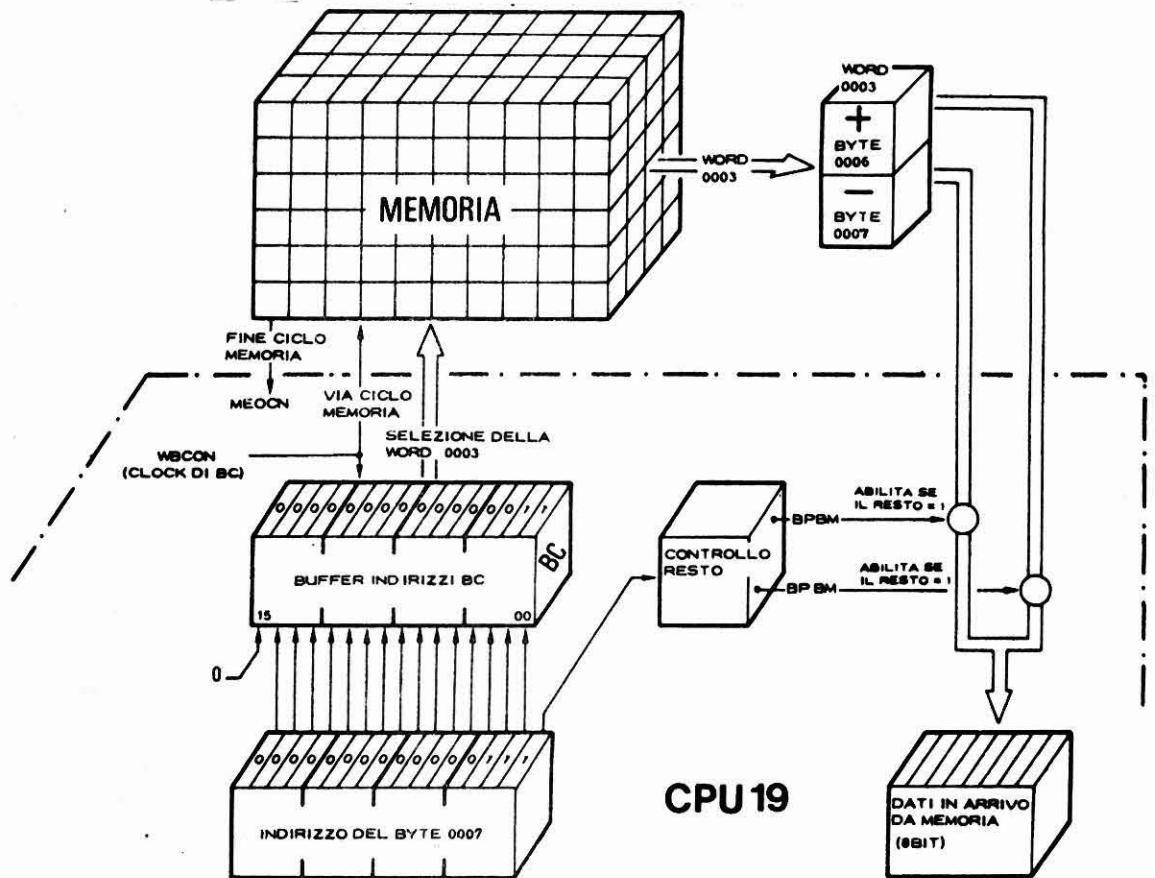


Fig. 11

MICROPROGRAM COUNTER

(Contatore di microprogramma)

Per eseguire un microprogramma occorre memorizzare in un registro di Scratch-pad, detto "Microprogram counter" l'indirizzo della microistruzione da eseguire.

Ad ogni microistruzione letta nella memoria (ROM o RAM), il "Microprogram counter" viene aggiornato automaticamente di uno in modo che l'indirizzo in esso contenuto corrisponda alla microistruzione successiva.

In assenza di microistruzioni di salto condizionato o incondizionato si ha la lettura e quindi l'esecuzione di una microistruzione dopo l'altra.

Le microistruzioni di salto modificano il contenuto del "Microprogram counter" interrompendo così la successione ordinata delle microistruzioni.

INTERFACCIA VERSO PERIFERICHE

Il CPU 19 può essere collegato ad un massimo di 16 diversi GOP (governi per unità periferiche).

Ogni GOP può a sua volta pilotare un massimo di 8 periferiche di input e 8 periferiche di output (fig. 12).

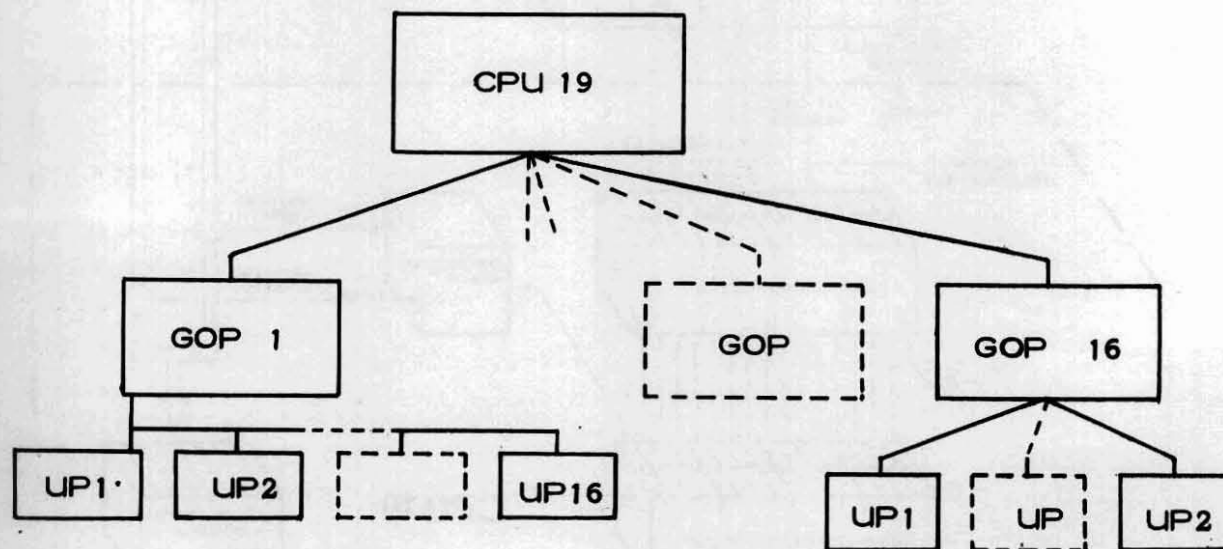


Fig. 12

Sovrapposizione fra governi

I GOP possono essere pilotati in sovrapposizione fra di loro : cioè è possibile che il CPU19 esegua in un certo periodo l'input o l'output (I/O) su uno o più governi.

Non è invece possibile la sovrapposizione tra due unità periferiche collegate ad uno stesso GOP.

La sovrapposizione degli I/O è resa possibile dal fatto che i tempi di lavoro del "processor" sono molto più brevi rispetto ai tempi di lavoro caratteristici delle periferiche.

Quest'ultime infatti necessitano di scambiare i dati o altre informazioni col processor (per periodi molto brevi), e poi azionano delle attuazioni ed eventualmente delle parti meccaniche impiegando tempi di ordine decisamente superiore.

Nel periodo in cui la periferica aziona le parti meccaniche od altre attuazioni, il processor dovrebbe restarsene in attesa e inutilizzato, invece lo si utilizza per procedere nel programma principale o per servire altre periferiche.

Il numero di GOP e quindi di periferiche collegabili al CPU19 contemporaneamente è funzione della frequenza con cui le periferiche richiedono l'intervento dell'unità centrale e del tempo che l'unità centrale impiega a servire le periferiche.

Interruzioni e livelli di priorità

Le periferiche (tramite i relativi GOP) interrompono il CPU19 per richiedere o fornire dati e informazioni in genere.

Quando è interrotto il CPU19 smette di eseguire il programma principale e si dedica a servire, per il tempo strettamente necessario, la periferica che ha interrotto.

Per ogni interruzione si ha uno scambio elementare di dati esempio : un byte, o di pochissime informazioni di altro genere.

Poichè i GOP che possono interrompere il CPU19 sono 16, nasce il problema di gestire delle code di interruzioni.

Ogni GOP può richiedere l'interruzione del CPU19 utilizzando uno dei tre fili a disposizione : EPR1, EPR2 e EPR3.

Le richieste di interruzione più urgenti vanno fatte col filo EPR1 e le altre su EPR2 o su EPR3 riservando al filo EPR3 le richieste di interruzioni meno urgenti.

Si hanno così tre diversi livelli di priorità nelle interruzioni e il CPU 19 darà la precedenza assoluta alle richieste di interruzione fatte tramite EPR1 e privilegiando le interruzioni fatte con EPR2 rispetto a quelle fatte con EPR3.

Il CPU 19 può lavorare in 4 assetti diversi (detti livelli):

- . Livello 4 o base : è il livello in cui il CPU 19 esegue il microprogramma principale o base; può essere interrotto con uno qualsiasi degli EPR.

Come microprogram counter il CPU usa L 0.

- . Livello 3 : è l'assetto che il CPU 19 assume quando è stato interrotto con il filo EPR3.

In questo assetto, il CPU 19 impedisce che altri governi possano generare EPR3, e quindi i governi che vogliono generare EPR3 dovranno attendere il ritorno a livello 4.

Il CPU 19 può essere interrotto con EPR2 e EPR1.

Come microprogram counter si usa L 1.

- . Livello 2 : è l'assetto che il CPU 19 assume quando è interrotto con EPR2.

Il CPU 19 impedisce che altri governi possano generare EPR3 o EPR2.

Il CPU 19 può essere interrotto con EPR1.

Come microprogram counter il CPU usa A 13.

- . Livello 1 : è l'assetto che il CPU 19 assume quando viene interrotto con EPR1, non può essere interrotto con nessuno degli EPR, perchè il CPU 19 li blocca tutti.

Come microprogram counter si usa A 12.

Riassumendo, si può concludere che in ognuno dei quattro livelli di lavoro il CPU 19 usa un particolare registro di scratch-pad per indirizzare il microprogramma e accetta interruzioni solo per passare a livelli più prioritari di quello in corso.

Trattamento delle interruzioni contemporanee

Abbiamo visto che il CPU 19, una volta passato a lavorare ad uno dei livelli di interruzioni : 1, 2 o 3, impedisce ai governi di generare l'EPR di richieste interruzione relative al livello in corso e al livello meno prioritario.

Questo impedimento però non può esistere nel caso che due GOP interrompano contemporaneamente e per risolvere questo caso si usano due sistemi diversi :

a) Sdoppiamento del livello 3 : Le richieste di interruzione di livello 3 possono essere fatte in realtà su due fili diversi : EPRA ed EPRB. Entrambi i fili determinano su CPU 19 il passaggio al livello 3. Se il CPU 19 è già a livello 3 non possono nascere richieste nè con EPRA nè con EPRB. Nel caso però di richieste contemporanee, vengono accettate quelle a livello 3A e messe in attesa quelle a livello 3B.

b) Priorità fisica

: Le richieste di interruzione contemporanee fatte coi fili EPR2, EPRA o EPRB vengono risolte col metodo della priorità fisica. Il CPU 19 accetta l'interruzione del livello più prioritario e segnala ai governi di essere passato al nuovo livello di lavoro. Dei due o più governi che abbiano interrotto al livello accettato dal CPU 19, uno sarà privilegiato dal cablaggio tra governi a ritenersi collegato con il CPU 19 e contemporaneamente lo stesso governo genera un segnale di blocco agli altri governi meno privilegiati.

E' quindi la posizione fisica della piastra GOP rispetto all'intercablaggio tra governi a risolvere il problema di interruzioni contemporanee.

MICROPROGRAMMI DI GESTIONE DELLE INTERRUZIONI

A seconda del livello di lavoro il CPU19 usa un diverso microprogram counter e ciò gli permette di poter eseguire quattro microprogrammi diversi (ovviamente in tempi diversi).

Mentre a livello 4 si esegue il microprogramma principale o base, ai livelli 3, 2 e 1 si eseguono tre microprogrammi per la gestione delle interruzioni da periferiche.

Vediamo ora con un esempio come viene gestita un'interruzione a livello 3 (fig. 13).

Nel momento in cui si esegue il microprogramma a livello 4, il registro $L\emptyset$ contiene l'indirizzo corrente del microprogramma base mentre $L1$ contiene l'indirizzo della 1° microistruzione del microprogramma di livello 3 (nell'esempio 8B04).

Al termine della microistruzione posta all'indirizzo 8A02 arriva una interruzione di livello 3, si cambia livello di lavoro, si indirizza il microprogramma con $L1$ mentre su $L\emptyset$ rimane l'indirizzo 8A03.

Viene così eseguito il microprogramma di livello 3, finchè una apposita microistruzione di rientro di livello (COM0) non determina il rientro a livello 4.

A questo punto si riprende il microprogramma di livello 4 dall'indirizzo rimasto su $L\emptyset$ e cioè 8A03 e su $L1$ rimane sempre l'indirizzo della prima microistruzione di livello 3 : 8B04.

MICROPROGRAMMA DI
livello 4

MICROPROGRAMMA DI
livello 3

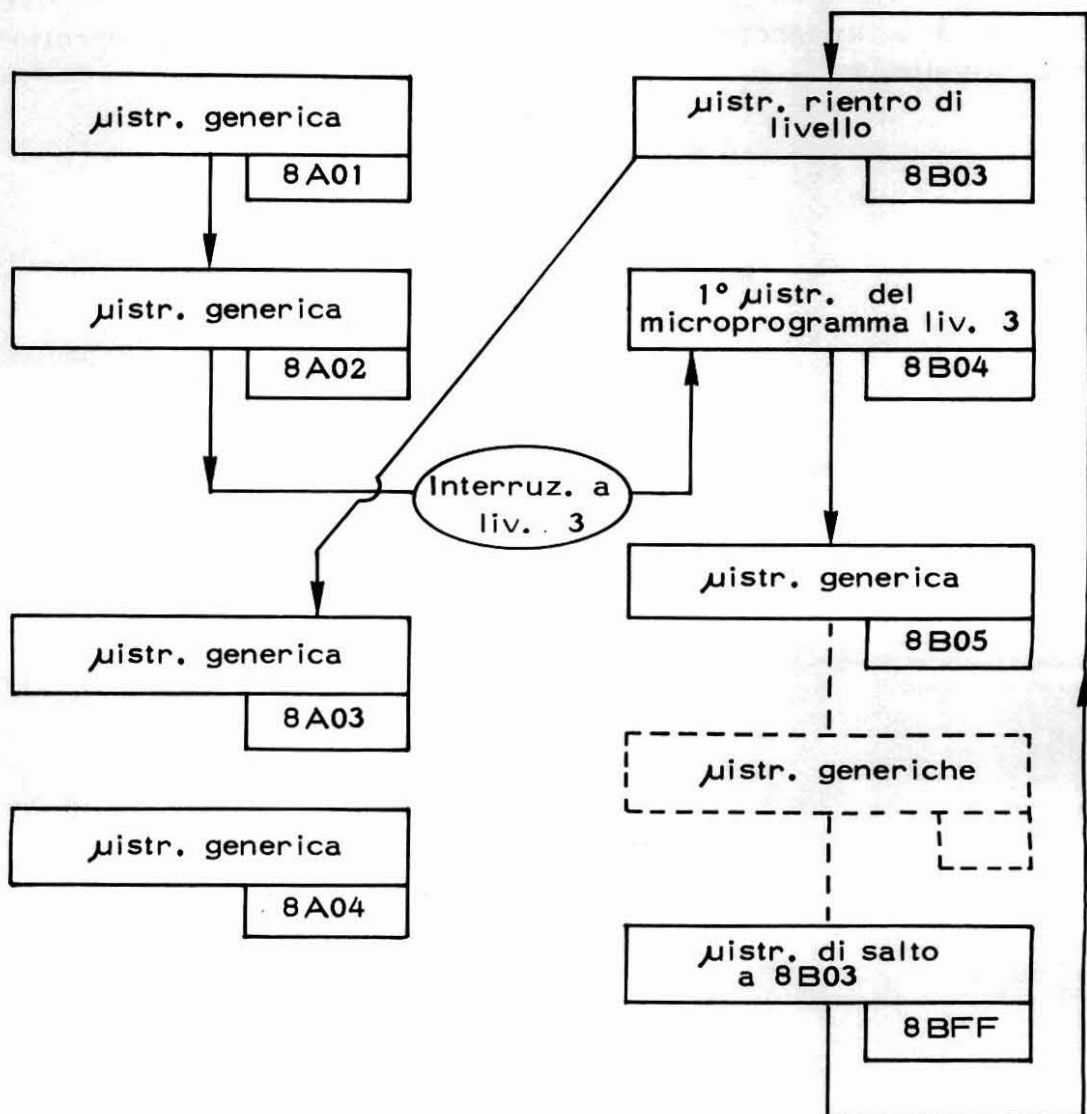
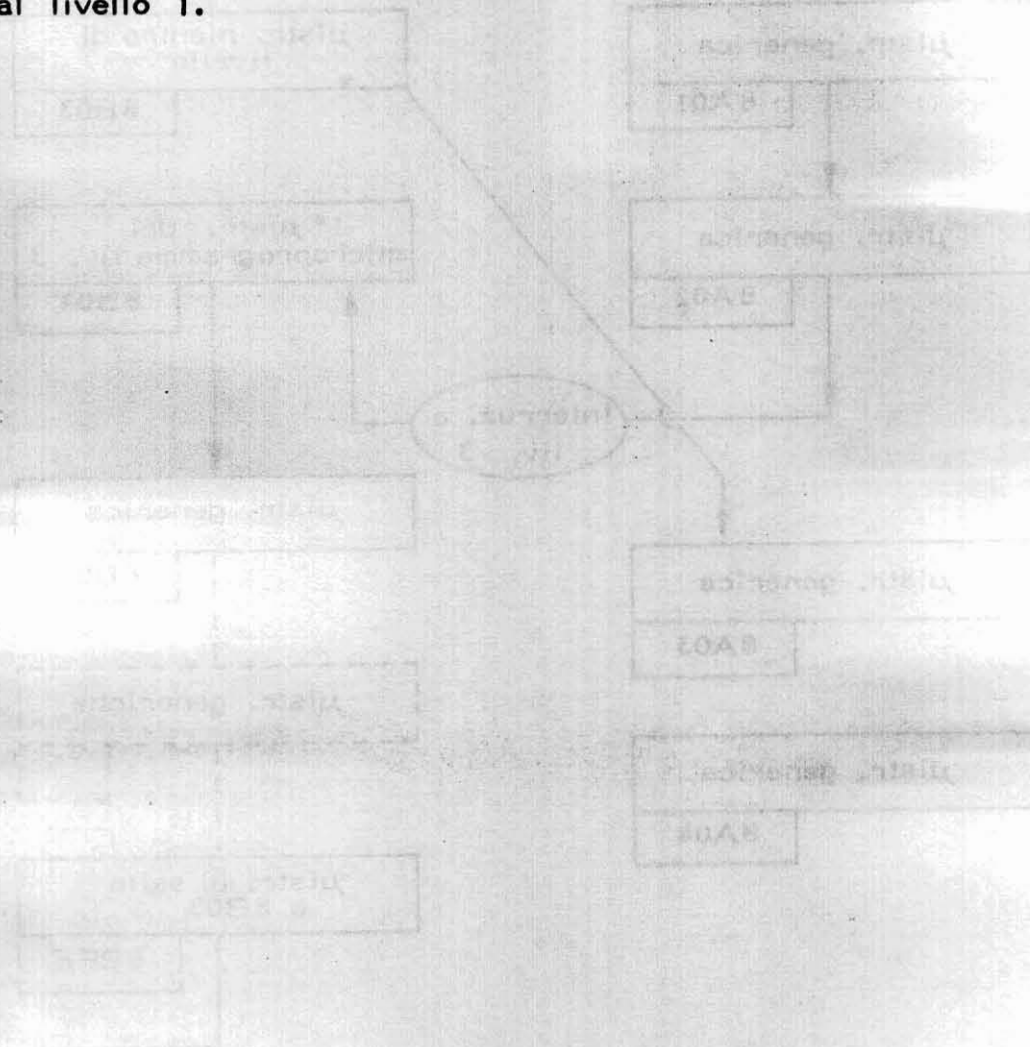


Fig. 13

In figura 14 abbiamo un altro esempio di interruzioni a livelli diversi: in particolare il livello 4 è interrotto a livello 3, il livello 3 a sua volta è interrotto a livello 2 e quest'ultimo è interrotto anche lui dal livello 1.



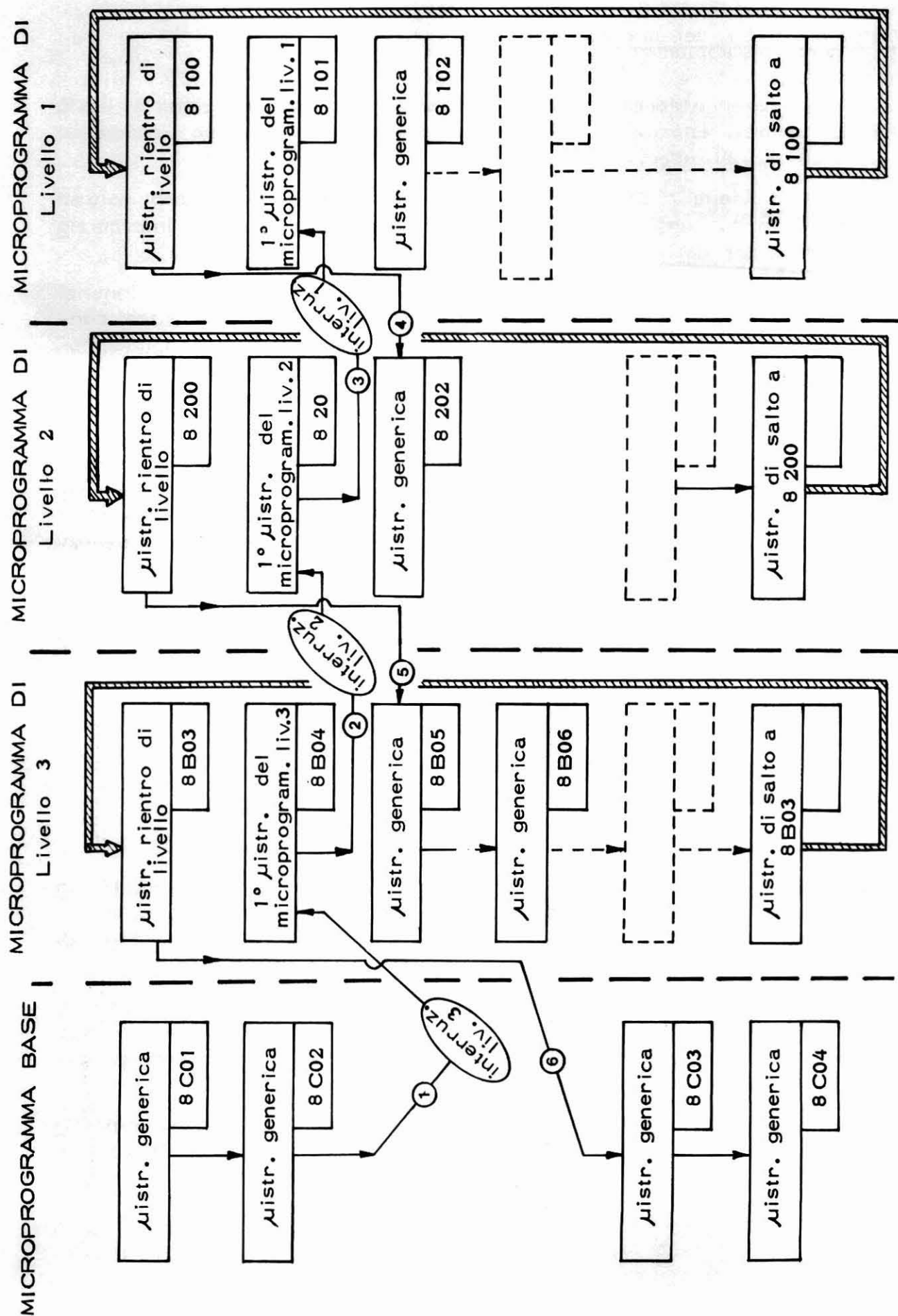


Fig. 14

Caratteristiche del microprogramma di livello 3

Dei tre microprogrammi per il trattamento delle interruzioni è il più lungo. La sua esecuzione richiede normalmente un tempo compreso tra i 60 e 120 μ sec.

Per questo viene richiesto da periferiche che se ne servono solo ad intervalli molto lunghi o comunque per scambi saltuari di informazioni diverse dai dati.

Per poter eseguire il microprogramma di livello 3 occorre innanzi tutto liberare alcuni registri di scratch-pad i quali senz'altro contengono delle informazioni necessarie allo svolgimento del microprogramma di livello 4. Quindi i dati tolti dallo scratch-pad devono essere salvati in memoria per poter poi riprendere correttamente il microprogramma di livello 4.

Poichè i GOP che possono interrompere a livello 3 sono 16 occorre chiedersi qual'è il governo che sta interrompendo e ciò può essere fatto richiedendo al governo di emettere su 8 fili "EPN0+7" il suo nome. Questo servirà ad indirizzare una zona di memoria riservata a tenere i parametri dell'I/O relativi al governo in questione.

I motivi per cui una periferica può interrompere a livello 3 sono diversi e affinché il CPU sappia qual'è il motivo dell'interruzione, il governo deve predisporre su 8 fili ("bus tipo" EPT0+7) il tipo di interruzione richiesto.

Riportiamo alcuni tipi di interruzioni a livello 3.

- Risposta ad una selezione da parte del CPU 19.
- Risposta ad un comando da parte del CPU 19.
- Fine di uno scambio dati.

Nella maggior parte dei casi è inoltre necessario che il CPU 19 conosca la condizione in cui si trova il governo prima di eseguire il microprogramma a livello 3 e questa è data su 8 fili detti "bus di stato".

Quando il CPU 19 conosce il nome della periferica che ha interrotto, le ragioni per cui ha interrotto e lo stato in cui si trova, preleva in memoria i parametri dell'I/O relativi alla periferica ed esegue il microprogramma specifico per quell'interruzione.

Quindi, prima di riprendere il microprogramma di livello 4, ripristina nello scratch-pad i dati relativi al microprogramma di livello 4 che aveva salvato in memoria.

Nel flusso di figura 15 viene riassunto il funzionamento del micro-programma di livello 3.

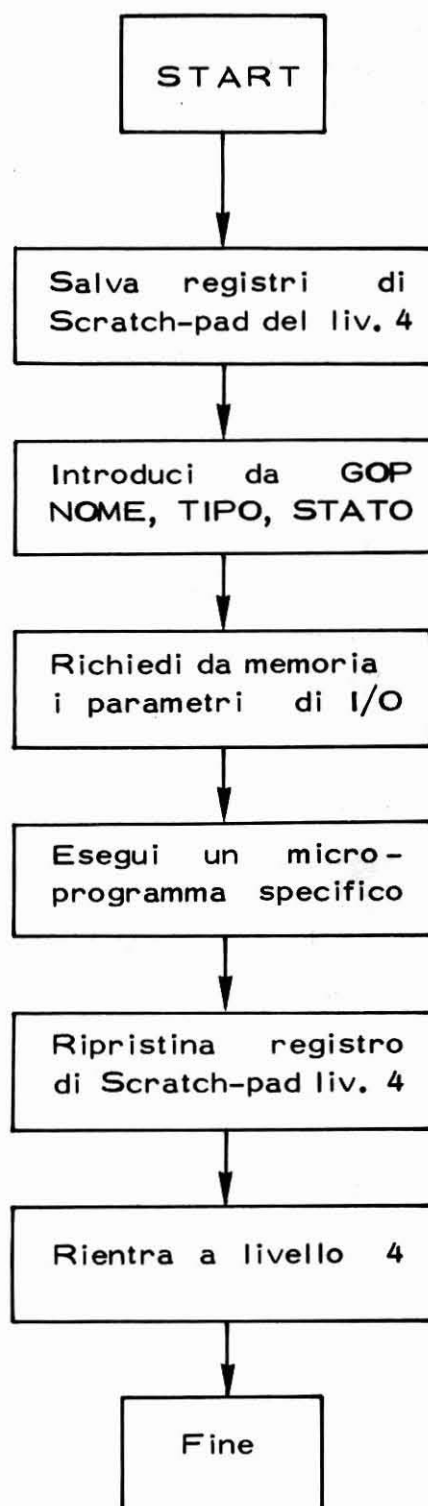


Fig. 15

Caratteristiche del microprogramma di livello 2

Le interruzioni a livello 2 sono permesse solo per lo scambio dati e non di informazioni generiche.

Poichè il microprogramma di livello 2 è specializzato per una operazione specifica la sua esecuzione è molto più veloce rispetto a quella del microprogramma di livello 3.

Il tempo medio richiesto per l'esecuzione del microprogramma di livello 2 è di circa $20 + 25$ microsecondi.

Durante le interruzioni a livello 2, i parametri necessari per l'I/O sono solo due :

- a) Numero di byte da scambiare con la periferica.
- b) Indirizzo del 1° byte di memoria da scambiare con la periferica.

Il microprogramma di livello 2 inizia sempre col salvataggio del contenuto di 3 registri di scratch-pad in memoria.

Poi si introduce il nome della periferica che ha interrotto e con questo si indirizza la memoria per prelevare i due parametri dell'I/O e portarli nello scratch-pad.

Si esegue quindi l'I/O di un byte, si aggiornano i parametri, sottraendo uno dal numero di byte da scambiare e sommando uno all'indirizzo del 1° byte da scambiare.

I parametri aggiornati vengono riportati in memoria e vengono ripristinati nello scratch-pad i dati dei tre registri che erano stati salvati in memoria e quindi si tronca definitivamente il microprogramma di livello 2.



Fig. 16

Caratteristiche del microprogramma di livello 1

Il livello 1 è il più specializzato dei tre livelli di interruzione; e i suoi limiti sono i seguenti:

- a) solo scambio dati e non informazioni generiche (come anche il microprogramma di livello 2) ;
- b) solo uno dei 16 governi può essere collegato in modo da interrompere a livello 1.

I due parametri di I/O di questo governo, anziché essere memorizzati in memoria come nel caso di governi che interrompono a livello 2, vengono memorizzati in due registri di scratch-pad riservati.

Questa specializzazione del livello 1 ha permesso di ridurre i tempi di esecuzione del microprogramma di livello 1 a soli 8 + 9 microsecondi.

Il livello 1 è quindi adatto a servire periferiche ad alto "transfer rate".

Il microprogramma a livello 1 è molto semplice, prevede solo scambio dati e l'aggiornamento dei parametri dell'I/O.

Essendo solo uno il governo che può interrompere a livello 1, non si chiede il nome della periferica, non si devono prelevare i parametri dalla memoria, poichè sono già nello scratch-pad e non occorre nè salvare nè ripristinare dei registri di scratch-pad.

Canali multiplex e canale selector (singolo)

Si dice che lavorano su canali multiplex le periferiche (o meglio i GOP) che eseguono lo scambio dati a livello 2 o 3: infatti queste periferiche possono lavorare in sovrapposizione fra di loro e quindi in modo multiplo.

La periferica (o GOP) che esegue lo scambio dati a livello 1 non può operare in sovrapposizione con periferiche dello stesso tipo anche se si sovrappone alle periferiche che operano sul canale multiplex o al microprogramma base; pertanto sono dette periferiche che operano su canale singolo o selector.

Canale D.M.A.

Il Direct Memory Access è un modo di collegare dei GOP in un sistema in modo tale da eseguire uno scambio dati diretto tra memoria e GOP senza passare attraverso il CPU 19.

Il Gop che opera in DMA gestisce in proprio i parametri dell'I/O relativi al 1° byte da scambiare e al numero di byte da scambiare e quando necessita dei dati indirizza direttamente la memoria, scambia i dati e aggiorna i parametri senza interrompere il CPU 19.

La prestazione DMA non è direttamente realizzabile sul CPU 19, ma lo diviene se si usano particolari piastre come ad esempio la RODMA.

In questo caso la memoria viene divisa in due parti (fig. 17):

- una riservata al solo CPU 19
- l'altra accessibile da CPU 19 e da GOP.

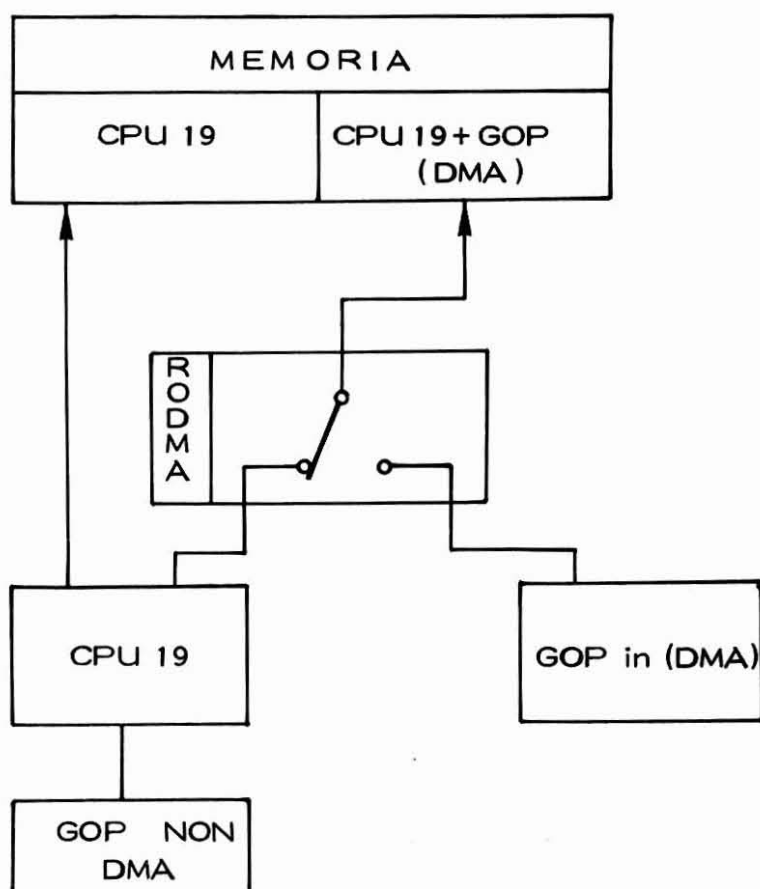
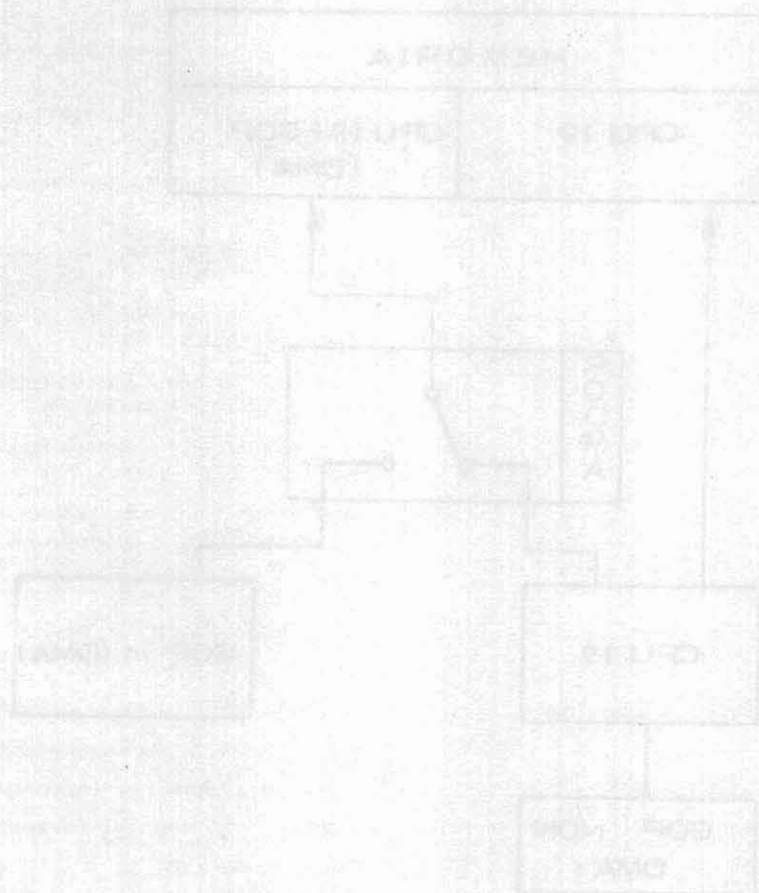


Fig. 17

In questo caso infatti il CPU19 può accedere alla memoria solo CPU19 in qualsiasi momento, mentre nella parte comune detta DMA può accedervi solo se in quel momento non è in atto uno scambio GOP Memoria.

Il GOP invece può accedere alla sola memoria DMA quando questa non è già impegnata dal CPU19.

La prestazione DMA è realizzata per servire le periferiche a più elevato transfer rate che sovraccaricherebbero eccessivamente il CPU19 anche se collegate sul canale singolo.



SUDDIVISIONE DELLE MICROISTRUZIONI PER FUNZIONE

MICROISTRUZIONI CHE OPERANO TRA REGISTRI	NOME	OPERAN- DO 1	OPE- RANDO 2	Cod. Esad.	DESCRIZIONE
Microistruzioni aritmetiche e logiche	ADDA	A(x)	B(y)	96xy	Addiziona $[Ax + By + D100]$ con risultato in Ax; posiziona i deviatori
	ADDB	A(x)	B(y)	A6xy	Addiziona $[Ax + By + D100]$ con risultato in By; posiziona i deviatori
	ADD	A(x)	B(y)	86xy	Addiziona $[Ax + By + D100]$ con risultato trascurato; posiziona i deviatori
	SOTA	A(x)	B(y)	C6xy	Sottrazione: $[Ax + \overline{By} + D100]$ con risultato in Ax; posiziona i deviatori
	SOTB	A(x)	B(y)	D6xy	Sottrazione: $[Ax + \overline{By} + D100]$ con risultato in By; posiziona i deviatori
	SQT	A(x)	B(y)	B6xy	Sottrazione: $[Ax + \overline{By} + D100]$ con risultato trascurato; posiziona i deviatori
	ANDA	A(x)	B(y)	A7xy	AND tra Ax e By con risultato in Ax; posiziona D101 se risultato = 0
	ANDB	A(x)	B(y)	B7xy	AND tra Ax e By con risultato in By; posiziona D101 se risultato = 0
	AND	A(x)	B(y)	97xy	AND tra Ax e By con risultato trascurato; posiziona D101 se risultato = 0
	ORA	A(x)	B(y)	F6xy	OR tra Ax e By con risultato in Ax; posiziona D101 se risultato = 0
	ORB	A(x)	B(y)	87xy	OR tra Ax e By con risultato in By; posiziona D101 se risultato = 0
	OR	A(x)	B(y)	E6xy	OR tra Ax e By con risultato trascurato; posiziona D101 se risultato = 0
	OREA	A(x)	B(y)	D7xy	OR esclusivo tra Ax e By con risultato in Ax; posiziona D101 se risultato = 0
	OREB	A(x)	B(y)	E7xy	OR esclusivo tra Ax e By con risultato in By; posiziona D101 se risultato = 0
	ORE	A(x)	B(y)	C7xy	OR esclusivo tra Ax e By con risultato trascurato; posiziona D101 se risultato = 0
Microistruzioni di trasferimento	TAB	A(x)	B(y)	D8xy	Trasferisce Ax in By
	TBA	A(x)	B(y)	E9xy	Trasferisce By in Ax
	TABM	A(x)	B(y)	E8xy	Trasferisce i 4 bit meno significativi di Ax in By
	TABP	A(x)	B(y)	D9xy	Trasferisce i 4 bit più significativi di Ax in By
	TBAM	A(x)	B(y)	F9xy	Trasferisce i 4 bit meno significativi di By in Ax
	TBAP	A(x)	B(y)	F8xy	Trasferisce i 4 bit più significativi di By in Ax
	SAB	A(x)	B(y)	BAxy	Scambio tra Ax e By
	SLL	L(x)	L(y)	BCxy	Scambio incrociato tra Lx e Ly

MICROISTRUZIONI CON MEMORIA	NOME	OPERAN DO 1	OPE- RANDO 2	Cod. Esad.	DESCRIZIONE
Microistruzioni con indirizzamento indiretto di memoria	MAI	$M_{06(x)} \leq 11$	A(y)	91xy	Memoria indirizzata da Mx (o Ax) in Ay
	MBI	$A_{126(x)} \leq 15$	B(y)	99xy	Memoria indirizzata da Mx (o Ax) in By
	MAIP	"	A(y)	98xy	Memoria indirizzata da Mx (o Ax) in Ay con incremento di 1 del contenuto di x
	MBIP	"	B(y)	9Cxy	Memoria indirizzata da Mx (o Ax) in By con incremento di 1 del contenuto di x
	MAIM	"	A(y)	92xy	Memoria indirizzata da Mx (o Ax) in Ay con decremento di 1 del contenuto di x
	MBIM	"	B(y)	9Axy	Memoria indirizzata da Mx (o Ax) in By con decremento di 1 del contenuto di x
	AMI	"	A(y)	A8xy	Ay in zona di memoria indirizzata da Mx (o Ax)
	BMI	"	B(y)	89xy	By in zona di memoria indirizzata da Mx (o Ax)
	AMIP	"	A(y)	88xy	A y in zona di memoria indirizzata da Mx (o Ax) con incremento di 1 del contenuto di x
	BMIP	"	B(y)	8Cxy	By in zona di memoria indirizzata da Mx (o Ax) con incremento di 1 del contenuto di x
	AMIM	"	A(y)	82xy	Ay in zona di memoria indirizzata da Mx (o Ax) con decremento di 1 del contenuto di x
	BMIM	"	B(y)	8Axy	By in zona di memoria indirizzata da Mx (o Ax) con decremento di 1 del contenuto di x
	MLI	"	L(y)	D1xy	Ly è caricato dal contenuto di memoria indirizza- ta da Mx (o Ax)
	MLIP	"	L(y)	DExy	Ly è caricato dal contenuto di memoria indirizza- ta da Mx (o Ax); incremento di 1 del contenuto di x
	MLIM	"	L(y)	DDxy	Ly è caricato dal contenuto di memoria indirizza- ta da Mx (o Ax); decremento di 1 del contenuto di x
	LMI	"	L(y)	E1xy	Ly in zona di memoria indirizzata da Mx (o Ax)
	LMIP	"	L(y)	EExy	Ly in zona di memoria indirizzata da Mx (o Ax) con incremento di 1 del contenuto di x
	LMIM	"	L(y)	EDxy	Ly in zona di memoria indirizzata da Mx (o Ax) con decremento di 1 del contenuto di x
	LPMIP	"	L(y)	E2xy	Incrementa di 1 Ly; il risultato viene inviato alla zona di memoria indirizzata da Mx (o Ax); incrementa di 1 il contenuto di x.
Microistruzioni con indirizzamento di- retto della memoria	MAD	A(x)	C(y)xy	3xyy	Deposita in Ax il contenuto della memoria all'indirizzo diretto Cyy
	AMD	A(x)	C(y)xy	2xyy	Deposita in memoria all'indirizzo diretto Cyy, il contenuto di Ax
Microistruzioni a caricamento imme- diato	CRTA	A(x)	C(y)xy	7xyy	Carica yy in Ax
	CRTB	B(x)	C(y)xy	5xyy	Carica yy in Bx
Microistruzioni di salto condizionato	SAD0	D(x)	nome simbo- lico	6dyy	$\left. \begin{array}{l} \text{Esegue il salto a yy se } Dx = 0 \\ \text{Esegue il salto a yy se } Dx = 1 \\ \text{Esegue il salto a xx se } ECOF0 = 1 \end{array} \right\} \alpha \text{ é la masche} \\ \text{ra del deviato-} \\ \text{re}$
	SADI	D(x)	"	6dyy	
	SADE	nome sim- bolico	"	4dxx	

	NOME	OPERAN- DO 1	OPE- RANDO 2	Cod. Esad.	DESCRIZIONE
Microistruzione di salto incondizionato	SAI	nome sim- bolico		0xxx 13 bit	Salto incondizionato a xxx
Microistruzioni tra memoria e governi di unità periferiche MEMORIA ↔ GOP	MEI	$\begin{cases} M_{04(x)411} \\ A_{124(x)415} \end{cases}$		90x0	Invia il byte di memoria indirizzata da Mx (o Ax) al canale dati ECD0 + 7
	MEIP	"		94x0	Invia il byte di memoria indirizzata da Mx (o Ax) al canale dati ECD0 + 7; incrementa di 1 il conte- nuto di x
	MEIM	"		9Dx0	Invia il byte di memoria indirizzata da Mx (o Ax) al canale dati ECD0 + 7; decrementa di 1 il conte- nuto di x
	EMI	"		8Dx8	Scriva EPD0 + 7 in memoria indirizzata da Mx (o Ax)
	EMIP	"		A2x8	Scriva EPD0 + 7 in memoria indirizzata da Mx (o Ax); incrementa di 1 il contenuto di x
	EMIM	"		A1x8	Scriva EPD0 + 7 in memoria indirizzata da Mx (o Ax); decrementa di 1 il contenuto di x
	ESE	"		B1x4	Emette la selezione del GOP (ECD0 + 7) da zona di memoria indirizzata da Mx (o Ax); emette ECOS ed ECOT
	ECO	"		B4x2	Emette il comando verso GOP (ECD0 + 7) da zona di memoria indirizzata da Mx (o Ax); emette ECOT ed ECOC
	SEI	"		F1x0	Emette ECD0 + F da zona di memoria indirizzata da Mx (o Ax); emette ECOT
	SEIP	"		F7x0	Emette ECD0 + F da zona di memoria indirizzata da Mx (o Ax); incrementa di 1 il contenuto di x ed emette ECOT
	SEIM	"		FDx0	Emette ECD0 + F da zona di memoria indirizzata da Mx (o Ax); decrementa di 1 il contenuto di x ed emette ECOT
	ESI	"		E0x8	Scriva EPD0 + 7 (o EPT0 + 7) in memoria indiriz- zata da Mx (o Ax); emette ECOT
	ESIP	"		EBx8	Scriva EPD0 + 7 (o EPT0 + 7) in memoria indiriz- zata da Mx (o Ax); incrementa di 1 il contenuto di x ed emette ECOT
	ESIM	"		ECx8	Scriva EPD0 + 7 (o EPT0 + 7) in memoria indiriz- zata da Mx (o Ax); decrementa di 1 il contenuto di x ed emette ECOT
MICROISTRUZIONI CHE OPERANO SU UN REGISTRO					
Microistruzioni di shift	SHSA	A(x)		C4x0	Shift a sinistra di Ax; il bit meno significativo è posto a 0 e il bit più significativo va su D100
	SHSB	B(x)		D4x0	Shift a sinistra di Bx; il bit meno significativo è posto a 0 e il bit più significativo va su D100
	SLSA	A(x)		C4x1	Shift ad anello, a sinistra di Ax; il bit più signifi- cativo va su D100 e il vecchio contenuto di D100 va sul bit meno significativo di Ax

	NOME	OPERAN- DO 1	OPE- RANDO 2	Cod. Esad.	DESCRIZIONE
Microistruzioni di shift	SLSB	B(x)		D4x1	Shift ad anello, a sinistra, di Bx; il bit più significativo va su D100 e il vecchio contenuto di D100 va sul bit meno significativo di Bx
	SHDA	A(x)		C3x0	Shift a destra di Ax; il bit meno significativo va su D100 e il bit più significativo è posto a 0
	SHDB	B(x)		D3x0	Shift a destra di Bx; il bit meno significativo va su D100 e il bit più significativo è posto a 0
	SLDA	A(x)		C3x1	Shift ad anello, a destra, di Ax; il bit meno significativo va su D100 e il vecchio contenuto di D100 va sul bit più significativo di Ax
	SLDB	B(x)		D3x1	Shift ad anello, a destra, di Bx; il bit meno significativo va su D100 e il vecchio contenuto di D100 va sul bit più significativo di Bx
Microistruzioni di incremento e decremento	ICA	A(x)		85xF	Incrementa di 1 il contenuto di Ax
	ICB	B(x)		95xF	Incrementa di 1 il contenuto di Bx
	ICL	L(x)		A5xF	Incrementa di 1 il contenuto di Lx
	DCA	A(x)		AExF	Decrementa di 1 il contenuto di Ax e posiziona D101: a 1 se il risultato = 0 a 0 negli altri casi
	DCB	B(x)		BExF	Decrementa di 1 il contenuto di Bx e posiziona D101: a 1 se il risultato = 0 a 0 negli altri casi
	DCL	L(x)		E5xF	Decrementa di 1 il contenuto di Lx e posiziona D101: a 1 se il risultato = 0 a 0 negli altri casi
	EDC	L(x)		ADxF	Decrementa di 1 il contenuto dei 12 bit meno significativi di Lx; quando tutti sono a 0 viene emesso ECOF
Microistruzioni di scambio di semi byte	ROTA	A(x)		8BxF	Ruota il contenuto di Ax
	ROTB	B(x)		9BxF	Ruota il contenuto di Bx
Microistruzioni di azzeramento	AZAP	A(x)		BBxF	Azzera i 4 bit più significativi di Ax
	AZBP	B(x)		DBxF	Azzera i 4 bit più significativi di Bx
	AZAM	A(x)		ABxF	Azzera i 4 bit meno significativi di Ax
	AZBM	B(x)		DBxF	Azzera i 4 bit meno significativi di Bx
Microistruzioni di verifica	VRA	A(x)		8ExF	Pone a 1 D101 se il contenuto di Ax è 0
	VRB	B(x)		9ExF	Pone a 1 D101 se il contenuto di Bx è 0
	VRL	L(x)		F5xF	Pone a 1 D101 se il contenuto di Lx è 0
MICROISTRUZIONI CHE OPERANO SUI DEVIATORI	TDIA	A(x)		C5xF	Trasferisce il contenuto dei deviatori in Ax
	TDIB	B(x)		D5xF	Trasferisce il contenuto dei deviatori in Bx
	TADI	A(x)		83xF	Trasferisce Ax nei deviatori
	TBDI	B(x)		93xF	Trasferisce Bx nei deviatori
	SDIA	A(x)		A3xF	Scambia il contenuto dei deviatori con Ax

	NOME	OPERAN- DO 1	OPE- RANDO 2	Cod. Esad.	DESCRIZIONE
MICROISTRUZIONI CHE OPERANO SUI DEVIATORI	SDIB	B(x)		B3xF	Scambia il contenuto dei deviatori con Bx
	SEDI	C(x)(x)		C9xx	Setta i deviatori mascherati da Cxx
	REDI	C(x)(x)		C8xx	Resetta i deviatori mascherati da Cxx
MICROISTRUZIONI TRA GOVERNI DI UNITA' PERIFERI- CHE E REGISTRI (GOP → Registri)	EDA	A(x)		B8x8	Deposita il canale dati EPD0 + 7 in Ax
	EDB	B(x)		A9x8	Deposita il canale dati EPD0 + 7 in Bx
	ENUA	A(x)		B9x0	Deposita il canale nome EPN0 + 7 in Ax
	ETIB	B(x)		B2xF	Deposita il canale tipo EPT0 + 7 in Bx
	ENTL	L(x)		AAx0	Deposita il canale nome EPN0 + 7 in Ax e il cana- le tipo EPT0 + 7 in Bx
MICROISTRUZIONI TRA REGISTRI E GOVERNI DI UNITA' PERIFERICHE (Registri → GOP)	DAE	L(x)		FCx0	L(x) viene emesso sul canale dati ECD0 + F accompagnato da ECOT
	CAE	L(x)		FCx2	L(x) viene emesso 2 volte sul canale dati ECD0 + F; la 2ª volta è accompagnata da ECOT ed ECOC
	DEA	L(x)		FBx8	Emette Bx su ECD8 + F accompagnato da ECOT e deposita EPD0 + 7 in Ax
MICROISTRUZIONI DI CONSOLLE (SUCE 2)	TCCA	A(x)		CAx0	Deposita gli 8 bit di consolle CON0 + 7 in Ax
	TDMA	A(x)		DAx1	Trasferisce i dati meno significativi di consolle da CON0 + 7 in Ax
	TDPA	A(x)		EAx2	Trasferisce i dati più significativi di consolle da CON0 + 7 in Ax
	TABC	A(x)	B(y)	FAxy	Trasferisce Ax e By in consolle accompagnato da C010
MICROISTRUZIONI DI COMANDO	COM0			BD00	Emette CE10N: disattiva il livello in corso, pone a 1 ECORN, non lavora a livello 4
	COM1			BD10	Emette CE11N: attiva il livello 3
	COM3			BD30	Emette CE13N: pone a 0 ECORN resettando i governi periferici
	COM4			BD40	Emette CE14N: fa nascere ECOT
	COM5			BD50	Emette CE15N
	COM6			BD60	Emette CE16N
	COM7			BD70	Emette CE17N: setta EC1F
	COM8			BD80	Emette CE18N: resetta EC1F
	COM9			BD90	Emette CE19N: setta EC2F
	COM10			BDA0	Emette CE20N: resetta EC2F
	COM11			BDB0	Emette CE21N: oscura i display del SUCE2
	COM12			BDC0	Emette CE22N: resetta il bit di start nel SUCE2
	COM13			BDD0	Emette CE23N: nel SUCE 2 segnala lo stop di ALT INDIRIZZATO
	COM14			BDE0	Emette CE24N: predispone un indirizzo di ALT da SUCE2
	COM15			BDF0	Emette CE25N: predispone il SUCE2 a visualizza- re sul display "DATA" il dato emesso da UC con una TABC.

	NOME	OPERAN- DO 1	OPE- RANDO 2	Cod. Esad.	DESCRIZIONE
PSEUDO MICROISTRUZIONI	ALFA	Registro indirizza- tore di mi- cropro- gramma: L00→liv. 4 L01→liv. 3 A13→liv. 2 A12→liv. 1			Non ha codice esadecimale in quanto è caricata su R0 da hardware. Indirizza la memoria per la lettura di una microistruzione e incrementa di 1 il registro indirizzatore di microprogramma.
	RESE	Registro indirizza- tore di micropro- gramma di livello 3: L01			Non ha codice esadecimale in quanto è caricata su R0 da hardware. Simula le condizioni di fine fase α e carica in L01 la prima microistruzione di inizializzazione (indirizzo esadecimale 8000).
	NOP			C900	Non opera: in fase α incrementa di 1 il registro indirizzatore del microprogramma in esecuzione: L00, L01, A13, A12.